33장 7번째 데이터 타입 Symbol

심벌

심벌(Symbol)은 ES6에서 도입된 7번째 데이터 타입으로 변경 불가능한 원시 타입의 값

주로 이름의 충돌 위험이 없는 유일한 프로퍼티 키를 만들기 위해 사용

심벌 값의 생성

심벌 값은 Symbol 함수를 호출하여 생성

```
// Symbol 함수를 호출하여 유일무이한 심벌 값을 생성한다.
const mySymbol = Symbol();
console.log(typeof mySymbol); // symbol
```

다른 값과 절대 중복되지 않는 유일무이한 값

```
new Symbol(); // TypeError: Symbol is not a constructor
```

심벌 값은 변경 불가능한 원시 값이다.

```
const mySymbol1 = Symbol('mySymbol');
const mySymbol2 = Symbol('mySymbol');

console.log(mySymbol1 === mySymbol2); // false
```

심벌 값에 대한 설명이 같더라도 생성된 심벌 값은 유일무이한 값

Symbol.for / Symbol.keyFor 메서드

수로 전달받은 문자열을 키로 사용하여

키와 심벌 값의 쌍들이 저장되어 있는 전역 심벌 레지스트리에서 해당 키와 일치하는 심벌 값을 검색

```
// 전역 심벌 레지스트리에 mySymbol이라는 키로 저장된 심벌 값이 없으면 새로
// 전역 심벌 레지스트리에 mySymbol이라는 키로 저장된 심벌 값이 있으면 해당
const s1 = Symbol.for('mySymbol');
const s2 = Symbol.for('mySymbol');

console.log(s1 === s2); // true
```

Symbol 함수는 호출될 때마다 유일무이한 심벌 값을 생성

Symbol.keyFor 메서드를 사용하면 전역 심벌 레지스트리에 저장된 심벌 값의 키를 추출

```
// 전역 심벌 레지스트리에 mySymbol이라는 키로 저장된 심벌 값이 없으면
새로운 심벌 값을 생성
const s1 = Symbol.for('mySymbol');
// 전역 심벌 레지스트리에 저장된 심벌 값의 키를 추출
Symbol.keyFor(s1); // mySymbol
```

심벌과 상수

```
// 위, 아래, 왼쪽, 오른쪽을 나타내는 상수를 정의한다.
// 이때 값 1, 2, 3, 4에는 특별한 의미가 없고 상수 이름에 의미가 있다.
const Direction = {
  UP: 1,
  DOWN: 2,
  LEFT: 3,
  RIGHT: 4
};

// 변수에 상수를 할당
const myDirection = Direction.UP;

if(myDirection === Direction.UP) {
  console.log('You are going UP.');
}
```

상수 이름 자체에 의미가 있는 경우

상수 값 1, 2, 3, 4가 변경될 수 있으며, 다른 변수 값과 중복될 수도 있다는 것는 문제가 될수 있다.

중복될 가능성이 없는 유일무이한 심벌 값을 사용

```
const Direction = {
  UP: Symbol('up'),
  DOWN: Symbol('down'),
  LEFT: Symbol('left'),
  RIGHT: Symbol('right')
};

const myDirection = Direction.UP;

if(myDirection === Direction.UP) {
  console.log('You are going UP.');
}
```

심벌과 프로퍼티 키

```
const obj = {
  // 심벌 값으로 프로퍼티 키를 생성
  [Symbol.for('mySymbol')]: 1
});

obj[Symbol.for('mySymbol')]; // 1
```

심벌 값은 유일무이한 값이므로 심벌 값으로 프로퍼티 키를 만들면 다른 프로퍼티 키와 절대 충돌하지 않는다.

기존 프로퍼티 키와 충돌하지 않는 것은 물론,

미래에 추가될 어떤 프로퍼티 키와도 충돌할 위험이 없다.

심벌과 프로퍼티 은닉

```
const obj = {
  // 심벌 값으로 프로퍼티 키를 생성
  [Symbol('mySymbol')]: 1
};

for (const key in obj) {
  console.log(key); // 아무것도 출력되지 않는다.
}

console.log(Object.keys(obj)); // []
console.log(Object.getOwnPropertyNames(obj)); // []
```

S6에서 도입된 Object.getOwnpropertySymbols 메서드를 사용하면

심벌 값을 프로퍼티 키로 사용하여 생성한 프로퍼티를 찾을 수 있다.

```
const obj = {
  // 심벌 값으로 프로퍼티 키를 생성
  [Symbol('mySymbol')]: 1
};

// getOwnPropertySymbols 메서드는 인수로 전달한 객체의 심벌 프로퍼
티 키를 배열로 반환한다.
console.log(Object.getOwnPropertySymbols(obj)); // [Symbol
(mySymbol)]

// getOwnPropertySymbols 메서드로 심벌 값도 찾을 수 있다.
const symbolKey1 = Object.getOwnPropertySymbols(obj)[0];
console.log(obj[symbolKey1]); // 1
```

Well-known Symbol

자바스크립트가 기본 제공하는 빌트인 심벌 값